REMARKS

Claim 1, as well as claims 14, 17 and 20, were amended to clarify that the branching operation is performed based on a byte, specified by the branching instruction, in a register, specified by the branch instruction, being equal or not equal to a byte value, also specified by the branch instruction, if the specified byte matches or mismatches the specified byte value. Support for this clarification is found, for example, at page 18, line 8, to page 20, line 11. In addition, claim 14 was amended to remove an unnecessary limitation from the preamble, and for greater clarity. Claims 16, 18, and 21 were also amended for greater clarity. Further, claims 11 and 12 were amended to correct an antecedent basis error, and claim 19 was amended to correct a dependency error and for greater clarity. After these amendments, claims 1-21 are pending. Claims 1, 14, 17, and 20 are independent.

The examiner rejected claims 1-13 under 35 U.S.C. §101 on the ground that the invention is not limited to tangible embodiments.

In the interest of advancing prosecution, applicant amended independent claim 1 to recite that the claim is directed to a method of operating a processor. Thus, applicant's independent claim 1 is directed to tangible embodiments of the invention, namely, a method of operating a processor which is a tangible item.

Additionally, applicant amended the preamble of claims 2-13 to recite that the claims are, like independent claim 1, directed to a method of operating the processor.

Applicant contends that the examiner rejection under 35 U.S.C. §101 has been overcome.

The examiner rejected claims 1-3, 6, 10, 13-21 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 4,724,521 to Carron et al, in view of U.S. Patent No. 5,802,373 to Yates.

The Examiner also rejected Claims 4 and 5 under 35 U.S.C. 103(a) as being unpatentable over Carron and Yates, in view of U.S. Patent No. 5,898,866 to Atkins et al.

The Examiner also rejected claims 7, 8, and 11 under 35 U.S.C. § 103(a) as being unpatentable over Carron and Yates, in view of U.S. Patent No. 4,742,151 to Bruckert et al.

The Examiner also rejected claim 9 under 35 U.S.C. § 103(a) as being unpatentable over Carron and Yates, in view of U.S. Patent No. 5,202,972 Gusefski et al.

The Examiner also rejected claim 12 under 35 U.S.C. § 103(a) as being unpatentable over Carron and Yates, in view of U.S. Patent No. 6,139,199 to Rodriguez.

Independent claim 1 recites, "executing a branch instruction that causes a processor to branch from executing a first sequential series of instructions to a different sequential series of instructions based on a byte, specified by the branch instruction, in a register, specified by the branch instruction, being equal or not equal to a byte value, specified by the branch instruction, if the specified byte matches or mismatches the specified byte value." Applicant's claimed method involves a branching instruction causing a comparison of a byte value, specified by the branching instruction, and the content of a byte of a register, also specified by the branching instruction. Based on the comparison, the branching instruction causes a branching operation to be performed. In other words, applicant's branch instruction causes both a comparison operation and a branching operation to be performed.

The examiner admits that "[h]owever, Carron does not explicitly teach of comparing a byte that is located in a register and performing branching instructions based on that comparison" (page 3 of the Office Action). Indeed, Carron in fact does not describe a compare or branching instruction of any sort, but rather describes branching and compare commands. As Carron explains:

> the method of this invention is based on storing in read only memory circuits within the local terminal a number of general purpose operation routines which comprise instructions to be executed by the central processor unit to accomplish a particular program task. **Each of these general purpose operation routines is associated with a defined command which, in its object code version, includes an operation code.** The object code version of the commands associated with the general purpose operation routines has a code length substantially shorter than the code length of the operation routine. In accordance with the method of this invention, the local terminal also has a program established in its read only memory system for interpreting the operation code to access the associated general purpose operation routine for execution by the central processor unit. The object code version of the application program in the form of a sequence of commands has a code size which is several times smaller than

the compiled code size would be for an entire application program which could be directly executed after downloading.

More specifically, the method of this invention includes a first step of establishing a set of general purpose operation routines to be executed by the local computer system. Each of these general purpose operation routines comprises a set of instructions for execution by the central processor unit in a prearranged manner to accomplish a specific task. The next step is to store the set of general purpose operation routines in the read only memory of the local terminal so that they may be accessed for execution by the central processor unit. The storage locations of these general purpose operation routines will be arranged and noted so that the routines can be addressed.

A following step is to define a set of commands, each of which is associated with a specific one of the general purpose operation routines. Each of the commands includes at least an operation code relating the command to its associated general purpose operation routine. Each of the commands is defined such that it has an associated command code length substantially less than the code length of the associated general purpose operation routine. Preferably, for convenience of writing programs, each command is defined with a high level programming syntax in which the command is represented by a series of word forms in abbreviated notation which have a recognizable association with the task that the associated general purpose operation routine will perform when it is executed by the central processor unit in the local terminal. This high level form of the command is then compiled and assembled to produce the operation code which relates to the associated general purpose operation routine in a manner which is intelligible to the interpreter program in the local terminal. (emphasis added, col. 7, line 59, to col. 8, line 47).

Carron defines commands that correspond to routines, stored in memory, each of which comprises several processor-executable instructions. One of those commands, listed in Carron's "SECTION FOUR: ALPHABETIC LISTING OF COMMANDS", is the COMP_BYTE command. Carron's COMP_BYTE, much like Carron's other commands, is not a single processor-executable instruction, but rather a command that invokes a routine comprising several instructions stored in memory. Invoking the COMP_BYTE command routine causes a byte at a specified position in a designated buffer to be compared with a byte constant in the command, and if the test passes (presumably, if the values of the compared operands match) the COMP_BYTE command routine causes branching operation to be performed (col. 134, lines 30-34).

Accordingly, Carron does not disclose or suggest "executing a branch instruction that causes a processor to branch from executing a first sequential series of instructions to a different sequential series of instructions based on a byte, specified by the branch instruction, in a register, specified by the branch instruction, being equal or not equal to a byte value, specified by the branch instruction, if the specified byte matches or mismatches the specified byte value," as required by applicant's independent claim 1.

The examiner also contended that: "Yates teaches of executing branch instructions based on comparing a byte in a register with a specified value (Col. 77, lines 29-33)" (paragraph 9 at page 3 of the Office Action).

Yates describes a computer system for executing a binary image conversion which converts instructions from an instruction set of a first non-native system to a second native computer system (Abstract). Amongst the operations that Yates' system performs is condition code processing in transformer (described at col. 76, line 11 to col. 78, line 34). One such condition code processing sequence is shown in Yates' FIG. 65B. As explained by Yates, the source instruction sequence 884 shown in FIG. 65B includes source instruction 884a, which performs a byte compare of register AL to the constant 3, and instruction 884b, which performs a branch if the value contained in the register AL is not equal to 3 (col. 77, lines 29-33).

In contrast to applicant's claimed method, Yates requires two separate instructions to perform the branching operation, namely, a compare instruction, followed by a separate conditional branch instruction. Further, as can be seen from FIG. 65B, the BNE instruction 884b does not specify any parameters or values, let alone a byte value and a byte of a register. At no point does Yates disclose executing a branch instruction that causes the processor to both perform a comparison of a byte value with the content of a byte of a register, and that then causes a branching operation to be performed based on that comparison. Accordingly, Yates does not disclose or suggests "executing a branch instruction that causes a processor to branch from executing a first sequential series of instructions to a different sequential series of instructions based on a byte, specified by the branch instruction, in a register, specified by the branch instruction, being equal or not equal to a byte value, specified by the branch instruction, if the

specified byte matches or mismatches the specified byte value," as required by applicant's independent claim 1.

Since neither Carron nor Yates discloses or suggests, alone or in combination, at least the features of "executing a branch instruction that causes a processor to branch from executing a first sequential series of instructions to a different sequential series of instructions based on a byte, specified by the branch instruction, in a register, specified by the branch instruction, being equal or not equal to a byte value, specified by the branch instruction, if the specified byte matches or mismatches the specified byte value," applicant's independent claim 1 is therefore patentable over the cited art.

Claims 2-13 depend from independent claim 1. Accordingly, claims 2-13 are patentable for at least the same reasons as independent claim 1.

Independent claims 14, 17 and 20 recite "a branch instruction that causes a processor to: fetch a byte, specified by the branch instruction, stored in a register, specified by the branch instruction; determine whether the byte in the register is equal or not equal to a specified byte value contained in the branch instruction; and perform a branching operation specified by the branch instruction based on the specified byte being equal or not equal to the byte in the register." For reasons similar to those provided with respect to independent claim 1, at least these features are not disclosed by the prior art. Accordingly, independent claims 14, 17, and 20 are patentable over the cited art.

Claims 15-16 depend from independent claim 14 and are therefore patentable for at least the same reasons as independent claim 14. Claims 18-19 depend from independent claim 17 and are therefore patentable for at least the same reasons as independent claim 17. Claim 21 depends from independent claim 20 and is therefore patentable for at least the same reasons as independent claim 20.

Additionally, as noted above, the examiner rejected claim 12 under 35 U.S.C. §103(a) as being unpatentable over Carron and Yates, in view of Rodriguez. Specifically, the examiner admits that "[a]s per claim 12, Carron does not teach the instruction of claim 1 [sic] wherein the branch instruction branches on a byte not matching the byte value and wherein the instruction

prefetches the next sequential instruction" (paragraph 36, page 9 of the Office Action). The examiner, however, contends that "Yates teaches of performing a branch instruction if the branch contained in the register is not equal to a specified value (col. 77, lines 29-33)" (paragraph 37, page 9 of the Office Action). Applicant disagrees.

Applicant's claim 12 describes "[t]he method of claim 1 wherein the branch instruction branches on the byte not matching the byte value and wherein the instruction prefetches the next sequential instruction."

As explained above, Yates discloses an instruction sequence that includes an instruction that performs a comparison operation whereby a byte of a register is compared to the value 3, and a separate instruction that performs a branching if the result of preceding comparison operation were such that the register byte was not equal to 3. Yates does not disclose that a branching instruction that performs both the comparison operation and the branching operation if the byte of the register, specified by the branch instruction, does not match the byte value specified by the branch instruction. Thus, Yates does not disclose or suggest "[t]he method of claim 1 wherein the branch instruction branches on the byte not matching the byte value and wherein the instruction prefetches the next sequential instruction," as required by applicant's claim 12.

Rodriguez describes a method for fast scheduling of instructions for just-in-time (JIT) compilers (col. 2, lines 57-63). While Rodriguez refers to branch instructions, nowhere does Rodriguez disclose or suggest executing a branch instruction that performs a comparison operation and branching operation that depends on the outcome of a comparison operation. Rodriguez certainly does not disclose a branch instruction that compares a byte value, specifies by the branch instruction, to a byte of a register, specified by the branch instruction, and branching on the byte value not matching the content of the byte of the register. Thus, Rodriguez does not disclose or suggest "[t]he method of claim 1 wherein the branch instruction branches on the byte not matching the byte value and wherein the instruction prefetches the next sequential instruction," as required by applicant's claim 12.

Applicant : Gilbert Wolrich et al.   Attorney's Docket No.: 10559-305US1 / P9626US
Serial No. : 10/069,229
Filed   : December 11, 2002
Page   : 13 of 13

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Please apply any charges to deposit account 06-1050, referencing attorney docket 10559-305US1.

<div style="text-align:center">Respectfully submitted,</div>

Date: Dec, 29, 20

Ido Rabinovitch
Reg. No. L0080

Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21236925.doc